PROJECT REPORT

Dynamic timetable generator

CSE2004 – DATABASE MANAGEMENT SYSTEMS

Submitted by

17BCE0833 – Shreyansh Kothari

17BCE0756 – Priyansh Agrawal

17BCE0372 – Vaibhav Raj Goel

17BCE0235 – Nakul Agrawal

Under the guidance of

Prof. R.Sathyaraj

Bachelor of Technology
in
Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

October 2018

Contents

1. ABSTRACT

The hand operated system of time table preparation in college is very monotonous and time-consuming which results in either same teacher ending up with more than one class or a number of classes conflicting at a same classroom. Due to Non-automated perspective absolute utilization of resources has proven ineffective. In order to deal with such problems, a mechanized system can be designed with a computer aided timetable generator. The system will take different inputs like number of subjects, teachers, maximum lectures a teacher can conduct, priority of subject, considering which, it will create feasible time tables for working days of the week, making excellent application of all resources in a way which will be best suited for the constraints. This work proposes an optimized technique to automate time table generation system. Time table generation system involves various challenging constraints of resources including faculties, rooms, time slots etc

2. INTRODUCTION

Even though most college administrative work has been computerized, the lecture timetable scheduling is still mostly done manually due to its inherent difficulties. The manual lecture-timetable scheduling demands considerable time and efforts. The lecture-timetable scheduling is a Constraint satisfaction problem in which we find a solution that satisfies the given set of constraints. A college timetable is a temporal arrangement of a set of lectures and classrooms in which all given constraints are satisfied. Creating such timetables manually is complex and time- consuming process. By automating this process with computer assisted timetable generator can save a lot of precious time of administrators who are involved in creating and managing course timetables. Since every college has its own timetabling problem, the commercially available software packages may not suit the need of every college. Hence the aim is to develop practical approach for building lecture course timetabling system, which can be customized to fit to any colleges timetabling problem. The college lecture-timetabling problem asks us to find some time slots and classrooms which satisfy the constraints imposed and offered. The lecture timetabling problem which essentially entails the assignment of rooms, students and teacher to a fixed time period, normally a working week.

3. LITERATURE SURVEY

Trying to develop a software which helps to generate Timetable for an Institution automatically. By looking at the existing system we can understand that timetable generation is done manually.

As we know all institutions) organisations have its own timetable, managing and maintaining these will not be difficult. Considering workload with this scheduling will make it more complex. As mentioned, when Timetable generation is being done, it should consider the maximum and minimum workload that is in a college. In those cases, timetable generation will become more complex. Also, it is a time-consuming process.

4. TOOLS AND METHODOLOGIES

Software and Hardware requirements

Platform forms the foundation on which the architecture, design, and implementation of a product is built. System specification defines the full functionality of the system. In many systems we work on, some functionality performed in hardware and some in software. System specification documents can thus be defined as the requirements documentation that formally specifies the system level requirements of an application. This application is developed in windows platform. We will be using PYTHON as our frontend language and MYSQL as our backend language for this software. Using MySQL in Python is what our main objective is to learn from this project.

MySQL (Back End)

MySQL is the worlds most used open source relational database Management system (RDBMS) as of IEEE that runs as a server providing multi-user access to a number of

databases. It is named after co-founder Michael Widenius's daughter. the phrase stands for structured query language. The MySQL development project has made its source code available under the terms of the terms of GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL lab, now owned by oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack LAMP is an acronym for Linux, Apache, MySQL, python/Perl/PHP " Free-software -open source projects that require a full-featured database management system often use MySQL.

Python (Front End)

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

External Python Libraries

XLRD – For reading excel sheets using python

XLWT – For writing excel sheet using python

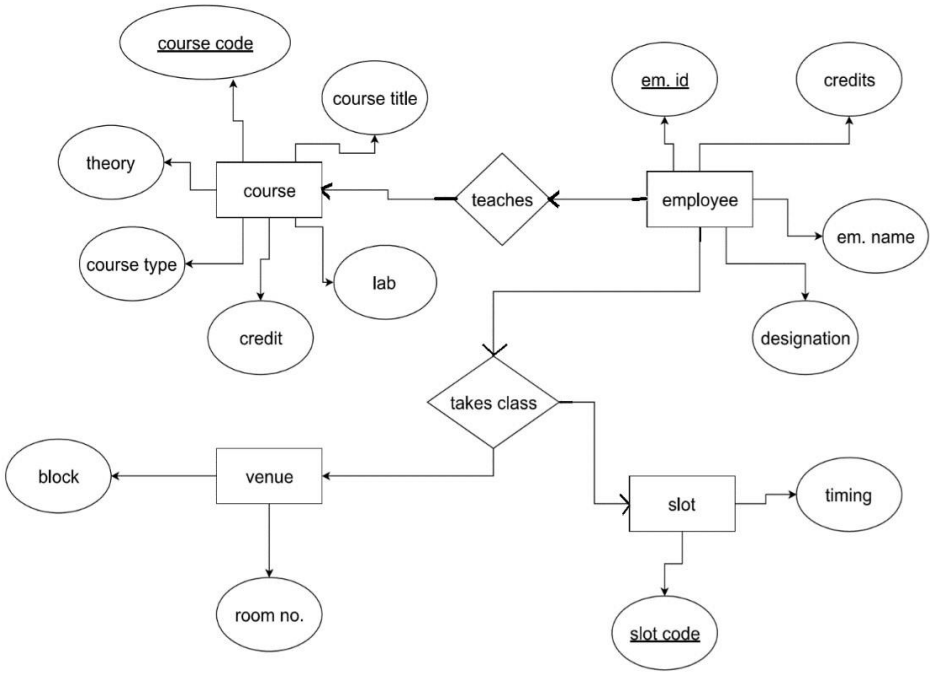
TKINTER – For GUI in python

PIL, XLUTILS

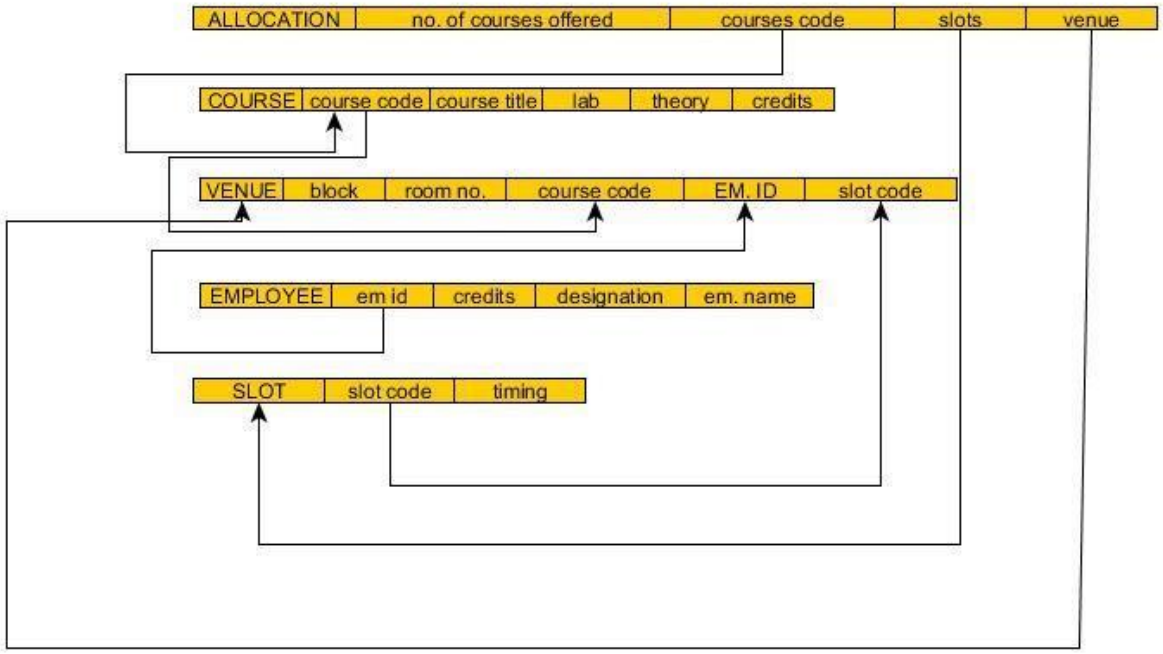
MS EXCEL

Microsoft Excel is a spreadsheet developed by Microsoft for Windows, macOS, Android and iOS. It features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications. Excel forms part of Microsoft Office. The files generated are in excel (.xls) format.

ER Diagram



Relational schema table



A design methodology is a methodical approach to creating a plan by applying a set of methods and guiding principle. We have followed these methodologies.

- Total requirement of the system including the framing of timetable strategy should be concerned. A database should be formed. As for every rule taken for the reason of maintaining the records. Record all possible scenarios and then upcoming with flow-charts to handle the scenario. The scheme should be carefully tested by running all the test cases written for the system.

- Firstly, we allow the concerned faculty to sign up and then login and then select 2 core and then 3 elective subjects and thus create a Wishlist for the same. Read the details like faculty, subjects, subject type into the database called the teacher_wishist as well as create a table course_type which contains name of the course, no of credits as well as the course type i.e. whether its elective or core subject.

- this process continues till all the teachers have chosen their core and elective subjects. the process can be ended even before all the teachers have chosen or not. After all the teaches have registered and chosen their subjects for the Wishlist. We move to creating the timetable and allocating the subjects to them

- Retrieve the table from the table wishlist_teachers and then we select teacher with the highest priorty and allocated their desired subjects to them and allocate them their Slot as well as the class room also note allocation strat from 3rd floor and in each floor there are 25 class rooms (301 to 325) where 315 to 319 are lab which wont be allocated to the
- After all the faculties have been allocated their classes and courses we allocate random courses to the faculties which didn't get the minimum number of subjects.
- After all the allocation are done and every teacher have been allocated at least 2 subjects we move on to creating the timetable.

5. Experiment and Result (CODE)

```
import mysql.connector
import xlrd
from xlwt import Workbook,Formula,easyxf
from tkinter import *
import os
from xlutils.copy import copy
from tkinter import ttk
from PIL import Image,ImageTk
import tkinter.messagebox
global e1
global e
global e2
global e5
```

```
global e6
global e3
global combon
global e4
global list1
global list2
list2=[["0","0"]]
list1=[["0","0","0","0"]]
global found
global insert
insert=[]
global m
global count
tt=mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="Viratrcb@1",
    database="timetable_generator"
)
cursor=tt.cursor(buffered=True)
#cursor.execute("create database timetable_generator")
#cursor.execute("use timetable_generator")

#course type table
print("\t\t\tWELCOME TO DYNAMIC TIME TABLE GENERATION\n")
print("connecting to mySql...\n")
print("1.creating course type table in database.....\n")
cursor.execute("drop table if exists course_type")
```

```
s="create table  course_type(course_name varchar(100) ,course_code varchar(10) PRIMARY KEY,credits integer(1) , core_batches integer(2), elective_batches integer(2),total_batches integer(2),type varchar(5),count_batches integer(2))"

cursor.execute(s)


#wishlist table

print("2.creating wishlist table in the database.....\n")

cursor.execute("drop table if exists teacher_wishlist")

s="create table teacher_wishlist(emp_name varchar(20) , emp_id integer(5),emp_desig varchar(30), emp_pass varchar(10),core_sub1 varchar(100),core_sub2 varchar(100),elec_sub1 varchar(100),elec_sub2 varchar(100),elec_sub3 varchar(100)) "

cursor.execute(s)

#opening teacher wishlist file (opening old file...change later)

teacher_wishlist_file="E:/VIT/Sem III/VIT Downloads/CSE2004-Database Management Systems/python project/data/teacher_details.xlsx"

teacher_wishlist=xlrd.open_workbook(teacher_wishlist_file)

teacher_wishlist_sheet=teacher_wishlist.sheet_by_index(0)

#s="truncate table teacher_wishlist"

#cursor.execute(s)

s="alter table teacher_wishlist add column desig_value integer(2)"

cursor.execute(s)

print("3.importing teacher wishlist excel data into teacher wishlist database table....\n")

designation=(("Senior Professor",1),("Professor",2),("Associate Professor",3),("Assistant Professor (SG)",4),("Assistant Professor (Sr)",5),("Assistant Professor",6))


"""s="drop table if exists teacher_details"

cursor.execute(s)

s="create table teacher_details(emp_name varchar(50),emp_is integer(5),emp_desig varchar(500),emp_pass varchar(15))"

cursor.execute(s)"""

sample=0

print("Enter 1 to use final wishlist file:\n")
```

```
sample=int(input())
for r in range(1,teacher_wishlist_sheet.nrows):
    name=teacher_wishlist_sheet.cell_value(r,0)
    emp_id=teacher_wishlist_sheet.cell_value(r,1)
    password=teacher_wishlist_sheet.cell_value(r,3)
    desig=teacher_wishlist_sheet.cell_value(r,2)
    s="insert into
teacher_wishlist(emp_name,emp_id,emp_desig)values('%s','%d','%s')"%(name,emp_id,desig
)
    cursor.execute(s)
    if(sample==1):
        core1=teacher_wishlist_sheet.cell_value(r,4)
        core2=teacher_wishlist_sheet.cell_value(r,5)
        elec1=teacher_wishlist_sheet.cell_value(r,6)
        elec2=teacher_wishlist_sheet.cell_value(r,7)
        elec3=teacher_wishlist_sheet.cell_value(r,8)
        values=(core1,core2,elec1,elec2,elec3)
        s="update teacher_wishlist set emp_pass='%s' where
emp_name='%s'"%(password,name)
        cursor.execute(s)
        s="update teacher_wishlist set
core_sub1='%s',core_sub2='%s',elec_sub1='%s',elec_sub2='%s',elec_sub3='%s' where
emp_id=%d"%(core1,core2,elec1,elec2,elec3,int(emp_id))
        cursor.execute(s)
    #s="update teacher_wishlist set emp_pass='%s' where emp_name='%s'"%(password,name)
    #list1.append([name,int(emp_id),desig,password])
    list1.append([name,int(emp_id),desig,"0"])
    for y in range(0,6):
        #print(desig,designation[y][0],designation[y][1])
        if(str(desig)==str(designation[y][0])):
            #print("entered")
```

```
        s="update teacher_wishlist set desig_value='%d' where
emp_name='%s'"%(int(designation[y][1]),name)
        cursor.execute(s)
        break

tt.commit()
print("4.importing course type excel file into course type database table...\n")
#opening course type file
course_type_file="E:/VIT/Sem III/VIT Downloads/CSE2004-Database Management
Systems/python project/data/course_type.xlsx"
course_type=xlrd.open_workbook(course_type_file)
course_type_sheet=course_type.sheet_by_index(0)

#entering value from course type file to table
for r in range(1,course_type_sheet.nrows):
    name=course_type_sheet.cell_value(r,0)
    code=course_type_sheet.cell_value(r,1)
    credit=course_type_sheet.cell_value(r,2)
    core=course_type_sheet.cell_value(r,3)
    elective=course_type_sheet.cell_value(r,4)
    values=(name,code,credit,core,elective)
    s="insert into
course_type(course_name,course_code,credits,core_batches,elective_batches)values('%s','%s'
,'%d','%d','%d')"%values
    cursor.execute(s)
    if(core>elective):
        s="update course_type set type='c' where course_code='%s'"%code
        cursor.execute(s)
    else:
        s="update course_type set type='e' where course_code='%s'"%code
        cursor.execute(s)
```

```
cursor.execute("update course_type set total_batches=core_batches+elective_batches")
cursor.execute("update course_type set count_batches=total_batches")


print("5.Creating DBMS PROJECT directory on the desktop...\n")
try:
    os.mkdir("C:/Users/shreyansh/Desktop/DBMS PROJECT")
except OSError:
    pass


#GUI START
def page1(d,m):
    for widget in m.winfo_children():
        widget.destroy()
    global list2
    global e1
    global e2
    global e4
    global e5
    global e6
    global e3
    global e
    if(int(d)!=1):
        tkinter.messagebox.showinfo("Successfull","Wishlist Submission and log out
Successful")
    def center_window(o,width=300, height=200):
        # get screen width and height
        screen_width = o.winfo_screenwidth()
        screen_height = o.winfo_screenheight()
```

```
        x = (screen_width/2) - (width/2)
        y = (screen_height/2) - (height/2)
        o.geometry('%dx%d+%d+%d' % (width, height, x, y))
    m.columnconfigure(0, weight=1)
    nb_of_columns = 2
    path="E:/VIT/Sem III/VIT Downloads/CSE2004-Database Management Systems/python project/data/v2.jpg"
    img = ImageTk.PhotoImage(Image.open(path))
    panel=Label(m,image=img)
    panel.grid(row=0,column=0)
    titlelabel = Label(m, text="Teacher Wishlist Portal", fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))
    titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)
    l5=Label(m,text="User Id :")
    l6=Label(m,text="Password :")
    e6=Entry(m)
    e5=Entry(m,show="*")
    #l4.grid(columnspan=2)
    l5.grid(row=2, sticky = E)
    e6.grid(row=2,column=1)
    l6.grid(row=3,sticky = E)
    e5.grid(row=3,column=1)
    button7=Button(m,text="Exit to Sign-up page",width=25, activebackground='GREY', command=lambda:page(m))
    button4=Button(m,text="login",width=25, activebackground='GREY', command=lambda: login(e6,e5,m))
    button4.grid(columnspan=2)
    button7.grid(columnspan=2)
    l5.place(x=140,y=175)
    e6.place(x=230,y=175)
    l6.place(x=140,y=200)
```

```
    e5.place(x=230,y=200)
    button4.place(x=160,y=225)
    button7.place(x=160,y=255)
    m.mainloop()



class app():
    def __init__(self):
        self.root1 = Tk(className=" VIT teacher Wishlist Portal")
        global insert
        insert=["1","2","3","4","5","6","7","8","9","10"]
        self.core_1()
        path="E:/VIT/Sem III/VIT Downloads/CSE2004-Database Management
Systems/python project/data/v3.jpg"
        img = ImageTk.PhotoImage(Image.open(path))
        panel=Label(self.root1,image=img)
        panel.grid(row=0,column=0)
        self.root1.mainloop()


    def core_1(self):
        global found
        #self.root1.geometry('500x300')
        width=600
        height=400
        screen_width=self.root1.winfo_screenwidth()
        screen_height=self.root1.winfo_screenheight()
        x = (screen_width/2) - (width/2)
        y = (screen_height/2) - (height/2)
        self.root1.geometry('%dx%d+%d+%d' % (width, height, x, y))
        self.l9=Label(self.root1,text="Core Subject 1:",font = ('Comic Sans MS',14))
```

```
        self.l9.place(x=50,y=150)
        self.combo=ttk.Combobox(self.root1,state='readonly',width=70)
        self.root1.columnconfigure(0, weight=1)
        nb_of_columns = 2
        self.titlelabel = Label(self.root1, text="                                Teacher Wishlist Portal
id: %d"%int(found), fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))
        self.titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)
        self.combo.set('select')
        cursor.execute("select course_name from course_type where type='c'")
        h=[]
        for i in cursor:
            h.append(i[0])
        self.combo['values']=h
        self.combo.place(x=50,y=180)
        self.button10 = Button(self.root1,text = "Select and goto Core subject 2",font = ('Comic
Sans MS',10),width=25, activebackground='GREY', command= self.core_2)
        self.combo.bind("<<ComboboxSelected>>",self.hi)
        self.button10.place(x=50,y=250)


    def core_2(self):
        global found
        #self.root1.destroy()
        self.l9.destroy()
        self.combo.destroy()
        self.button10.destroy()
        self.l9=Label(self.root1,text="Core Subject 2:",font = ('Comic Sans MS',14))
        self.l9.place(x=50,y=150)
        self.root1.columnconfigure(0, weight=1)
        nb_of_columns = 2
        self.titlelabel = Label(self.root1, text="                                Teacher Wishlist Portal
id: %d"%int(found), fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))
```

```
        self.titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)
        self.combo=ttk.Combobox(self.root1,state='readonly',width=70)
        self.combo.set('select')
        cursor.execute("select course_name from course_type where type='c'")
        h=[]
        for i in cursor:
            global insert
            if(i[0]!=insert[0]):
                h.append(i[0])
        self.combo['values']=h
        self.combo.place(x=50,y=180)
        self.button10 = Button(self.root1,text = "Select and goto Elective subject 1",font =
('Comic Sans MS',10),width=25, activebackground='GREY', command=self.elective_1)
        self.combo.bind("<<ComboboxSelected>>",self.hi1)
        self.button10.place(x=50,y=250)
        self.root1.mainloop()


    def elective_1(self):
        global found
        #self.root3=Tk()
        self.combo.destroy()
        self.l9.destroy()
        self.button10.destroy()
        self.l9=Label(self.root1,text="Elective Subject 1:",font = ('Comic Sans MS',14))
        self.l9.place(x=50,y=150)
        self.combo=ttk.Combobox(self.root1,state='readonly',width=70)
        self.combo.set('select')
        self.root1.columnconfigure(0, weight=1)
        nb_of_columns = 2
```

```
        self.titlelabel = Label(self.root1, text="                                Teacher Wishlist Portal
id: %d"%int(found), fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))

        self.titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)

        cursor.execute("select course_name from course_type where type='e'")

        h=[]

        for i in cursor:

            h.append(i[0])

        self.combo['values']=h

        self.combo.place(x=50,y=180)

        self.button10 = Button(self.root1,text = "Select and goto Elective subject 2",font =
('Comic Sans MS',10),width=25, activebackground='GREY', command=self.elective_2)

        self.combo.bind("<<ComboboxSelected>>",self.hi2)

        self.button10.place(x=50,y=250)

        self.root1.mainloop()


    def elective_2(self):

        global found

        self.combo.destroy()

        self.button10.destroy()

        self.l9.destroy()

        self.l9=Label(self.root1,text="Elective Subject 2:",font = ('Comic Sans MS',14))

        self.l9.place(x=50,y=150)

        self.combo=ttk.Combobox(self.root1,state='readonly',width=70)

        self.combo.set('select')

        self.root1.columnconfigure(0, weight=1)

        nb_of_columns = 2

        self.titlelabel = Label(self.root1, text="                                Teacher Wishlist Portal
id: %d"%int(found), fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))

        self.titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)

        cursor.execute("select course_name from course_type where type='e'")

        h=[]
```

```
        for i in cursor:
            global insert
            if(i[0]!=insert[2]):
                h.append(i[0])
        self.combo['values']=h
        self.combo.place(x=50,y=180)
        self.button10 = Button(self.root1,text = "Select and goto Elective subject 3",font =
('Comic Sans MS',10),width=25, activebackground='GREY', command=self.elective_3)
        self.combo.bind("<<ComboboxSelected>>",self.hi3)
        self.button10.place(x=50,y=250)
        self.root1.mainloop()


    def elective_3(self):
        global found
        self.combo.destroy()
        self.button10.destroy()
        self.l9.destroy()
        self.l9=Label(self.root1,text="Elective Subject 3:",font = ('Comic Sans MS',14))
        self.l9.place(x=50,y=150)
        self.combo=ttk.Combobox(self.root1,state='readonly',width=70)
        self.combo.set('select')
        self.root1.columnconfigure(0, weight=1)
        nb_of_columns = 2
        self.titlelabel = Label(self.root1, text="                                    Teacher Wishlist Portal
id: %d"%int(found), fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))
        self.titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)
        cursor.execute("select course_name from course_type where type='e'")
        h=[]
        for i in cursor:
            global insert
```

```
            if(i[0]!=insert[2] and i[0]!=insert[3]):
                h.append(i[0])
        self.combo['values']=h
        self.combo.place(x=50,y=180)
        self.button10 =Button(self.root1,text = "Submit and log out",font = ('Comic Sans MS',10),width=25, activebackground='GREY', command=self.p1)
        self.combo.bind("<<ComboboxSelected>>",self.hi4)
        self.button10.place(x=50,y=250)
        self.root1.mainloop()



    def hi(self, event):
        self.x = self.combo.get()
        #print(self.x)
        s="select course_code from course_type where course_name='%s'"%(self.x)
        cursor.execute(s)
        global insert
        q=cursor.fetchone()
        insert[5]=q
        #insert=[]
        insert[0]=self.x


    def hi1(self, event):
        self.y = self.combo.get()
        #print(self.y)
        s="select course_code from course_type where course_name='%s'"%(self.y)
        cursor.execute(s)
        global insert
        q=cursor.fetchone()
        insert[6]=q
```

```
    insert[1]=self.y

def hi2(self, event):
    self.z =self.combo.get()
    #print(self.z)
    s="select course_code from course_type where course_name='%s'"%(self.z)
    cursor.execute(s)
    global insert
    q=cursor.fetchone()
    insert[7]=q
    insert[2]=self.z

def hi3(self, event):
    self.v =self.combo.get()
    #print(self.v)
    s="select course_code from course_type where course_name='%s'"%(self.v)
    cursor.execute(s)
    q=cursor.fetchone()
    global insert
    insert[8]=q
    insert[3]=self.v

def hi4(self, event):
    self.q =self.combo.get()
    #print(self.q)
    s="select course_code from course_type where course_name='%s'"%(self.q)
    cursor.execute(s)
    global insert
    z=cursor.fetchone()
```

```
        insert[9]=z
        insert[4]=self.q


    def p1(self):
        global insert
        #print(insert)
        global found
        s="update teacher_wishlist set
core_sub1='%s',core_sub2='%s',elec_sub1='%s',elec_sub2='%s',elec_sub3='%s' where
emp_id=%d"%(insert[0+5][0],insert[1+5][0],insert[2+5][0],insert[3+5][0],insert[4+5][0],int(f
ound))
        cursor.execute(s)
        page1(2,self.root1)



def login(userid ,password,root):
    global list2
    global count
    global e1
    global e2
    global e4
    global e3
    global e
    for i in range(len(list2)):
        if(userid.get()=="" or password.get()==""):
            count=0
            break
        elif str(userid.get().rstrip()) == str(list2[i][0]) and str(password.get().rstrip()) ==
str(list2[i][1]):
            count=1
            global found
```

```
            found=int(userid.get().rstrip())
            userid.delete(0,END)
            password.delete(0,END)
            break
        else:
            count = 0

    if (count == 1):
        s="select core_sub2 from teacher_wishlist where emp_id='%d'"%(int(list2[i][0]))
        cursor.execute(s)
        t=cursor.fetchall()
        if(str(t[0][0])!='None'):
            tkinter.messagebox.showerror("Error","Wishlist Already Submitted")
            userid.delete(0,END)
            password.delete(0,END)
        else:
            root.destroy()
            app()
    else:
        tkinter.messagebox.showerror("Error","Invalid Username or Password")
        userid.delete(0,END)
        password.delete(0,END)

def sign_up():
    global list2
    global list1
    global e1
    global e2
    global e4
```

```
    global combon
    global e3
    global e
    for i in range(len(list1)):
        if(e.get()=="" or e2.get()=="" or e3.get()==""):
            tkinter.messagebox.showerror("Error","The entries cannot be empty")
            count=2
            break
        elif int(e.get().rstrip()) == int(list1[i][1]) and e1.get().rstrip()==str(list1[i][0]) and
e4==str(list1[i][2]):
            count=1
            break
        else:
            count = 0
    if(count==1):
        if(e2.get().rstrip() == e3.get().rstrip()):
            for i in range(0,len(list2)):
                u=0
                if(int(e.get().rstrip())==int(list2[i][1])):
                    u=1
                    break
            if(u==1):
                tkinter.messagebox.showerror("Error", "Already Signed up")
                e.delete(0, END)
                e1.delete(0, END)
                e2.delete(0, END)
                e3.delete(0, END)
                combon.set('select')
            else:
                list1[i][3]=e2.get().rstrip()
```

```
                list2.append([int(e.get().rstrip()),e2.get().rstrip()])
                tkinter.messagebox.showinfo("Successfull","Sign-up Successfull")
                e.delete(0,END)
                e1.delete(0, END)
                e2.delete(0, END)
                e3.delete(0, END)
                combon.set('select')
            else:
                tkinter.messagebox.showerror("Error","Password Does Not Match")
                e2.delete(0, END)
                e3.delete(0, END)
        elif(count==0):
            tkinter.messagebox.showerror("Error", "Invalid Entries")
            e.delete(0, END)
            e1.delete(0, END)
            e2.delete(0, END)
            e3.delete(0, END)
            combon.set('select')

    tt.commit()


    def page(m):
        global l
        global e1
        global e2
        global e3
        global combon
        global e4
        global e
```

```
    for widget in m.winfo_children():
        widget.destroy()
    #m.configure(background='White')

    path="E:/VIT/Sem III/VIT Downloads/CSE2004-Database Management Systems/python project/data/v1.jpg"
    img = ImageTk.PhotoImage(Image.open(path))
    panel=Label(m,image=img)
    panel.grid(row=0,column=0)
    m.columnconfigure(0, weight=1)
    nb_of_columns = 2
    titlelabel = Label(m, text="Teacher Wishlist Portal", fg="blue4", bg ="gray80",font = ('Comic Sans MS',12))
    titlelabel.grid(row=5, column=0, sticky='ew', columnspan=nb_of_columns)
    l10=Label(m,text="User id :")
    l1=Label(m,text="User Name :")
    l4=Label(m,text="Designation :")
    def func(event):
        global e4
        e4=combon.get()
    combon=ttk.Combobox(m,state='readonly',width=20)
    combon.set('select')
    h=("Senior Professor","Professor","Associate Professor","Assistant Professor (SG)","Assistant Professor (Sr)","Assistant Professor")
    combon['values']=h
    combon.bind("<<ComboboxSelected>>",func)
    l2=Label(m,text="Create Password :")
    l3=Label(m,text="Confirm Password :")
    e=Entry(m)
    e1=Entry(m)
    e2=Entry(m,show="*")
```

```
    e3=Entry(m,show="*")
    combon.place(x=140,y=200)
    l4.place(x=20,y=200)
    l10.place(x=20,y=150)
    e.place(x=140, y=150)
    l1.place(x=20, y=175)
    e1.place(x=140,y=175)
    l2.place(x=20,y=225)
    e2.place(x=140,y=225)
    l3.place(x=20,y=250)
    e3.place(x=140,y=250)
    global count
    count=0
    button3=Button(m,text="Sign up",width=25, activebackground='GREY',
command=sign_up)
    button3.grid(columnspan=2)
    button3.place(x=45,y=300)
    button1=Button(m,text="Login",width=25, activebackground='GREY', command=lambda:
page1(1,m))
    button = Button(m, text = 'Exit', width=25, activebackground='GREY',
command=m.destroy)
    button1.grid(columnspan=2)
    button1.place(x=325,y=300)
    button.grid(columnspan=2)
    button.place(x=325,y=335)


    m.mainloop()


def GUISTART():
    m=Tk(className=" VIT Vellore")
    width=600
```

```
    height=400
    screen_width=m.winfo_screenwidth()
    screen_height=m.winfo_screenheight()
    x = (screen_width/2) - (width/2)
    y = (screen_height/2) - (height/2)
    m.geometry('%dx%d+%d+%d' % (width, height, x, y))
    page(m)

if(sample!=1):
    print("6.Initiating GUI....\n")
    GUISTART()
if(sample!=1):
    print("7.GUI exited....\n")
    print("enter the GUI again ?")
    print("Enter 'Y' for YES and 'N' for NO:")
    select=input()
    if(select=='Y' or select =='y'):
        GUISTART()

print("8.Creating Course Details table in database...\n")
#course details table
cursor.execute("drop table if exists course_details")
s="create table  course_details(course_name varchar(100) ,course_code varchar(10),slot
varchar(5), batches integer(2))"
cursor.execute(s)

print("9.Creating Allocation table in database....\n")
#allocation table
cursor.execute("drop table if exists allocation")
```

```
s="create table allocation(emp_name varchar(20),emp_id integer (5) ,course_name
varchar(100),course_code varchar(10),course_credits integer(1),slot varchar(10),venue
varchar(10))"

cursor.execute(s)


print("10.Importing Course Details excel file data into the table....\n")

#opening course details (slots) file

course_detail_file="E:/VIT/Sem III/VIT Downloads/CSE2004-Database Management
Systems/python project/data/course_details.xlsx"

course_detail=xlrd.open_workbook(course_detail_file)

course_detail_sheet=course_detail.sheet_by_index(0)

for r in range(1,course_detail_sheet.nrows):

    name=course_detail_sheet.cell_value(r,0)

    code=course_detail_sheet.cell_value(r,1)

    slot=course_detail_sheet.cell_value(r,2)

    batches=course_detail_sheet.cell_value(r,3)

    values=(name,code,slot,batches)

    s="insert into
course_details(course_name,course_code,slot,batches)values('%s','%s','%s','%d')"%values

    cursor.execute(s)

tt.commit()


print("11.Creating teacher wishlist excel file in DBMS PROJECT directory....\n")

#write wishlist file using table

teacher=Workbook()

sheet1=teacher.add_sheet('teacher wishlist')

style=easyxf('pattern:pattern solid,fore_colour red;align:horiz centre;')

sheet1.write(0,0,'NAME',style)

sheet1.write(0,1,'EMPLOYEE ID',style)

sheet1.write(0,2,'DESIGNATION',style)

#sheet1.write(0,3,'PASSWORD',style)
```

```
sheet1.write(0,4,'CORE 1',style)
sheet1.write(0,5,'CORE 2',style)
sheet1.write(0,6,'ELECTIVE 1',style)
sheet1.write(0,7,'ELECTIVE 2',style)
sheet1.write(0,8,'ELECTIVE 3',style)
sheet1.col(0).width=5000
sheet1.col(1).width=4000
sheet1.col(2).width=6000
sheet1.col(3).width=4000
sheet1.col(8).width=4000
sheet1.col(4).width=4000
sheet1.col(5).width=4000
sheet1.col(6).width=4000
sheet1.col(7).width=4000
print("12.Exporting final teacher wishlist table to a teacher wishlist excel file...\n")
s="select * from teacher_wishlist"
cursor.execute(s)
r=cursor.fetchall()
p=len(r)
x=0
c=1
while(c<=p):
    x=0
    while(x<9):
        if(x!=3):
            sheet1.write(c,x,r[c-1][x])
            x=x+1
        else:
            x=x+1
```

```
    c=c+1
teacher.save('C:/Users/shreyansh/Desktop/DBMS PROJECT/teacher_wishlist.xls')
"""select=input("input: ")
if(int(select)==1):
    fe()
#elif(int(select)==2):
else:
    tab()"""
tt.commit()
cursor.execute("drop table if exists teacher_wishlistc")
cursor.execute("create table teacher_wishlistc as select * from teacher_wishlist")
cursor.execute("alter table teacher_wishlistc add column ind integer(2)")

print("13.STARTING ALLOCATION\n")

print("14.Allocating maximum of 2 subject based of faculty priority....\n")
#first 2 subjects allocation
cursor.execute("select * from teacher_wishlistc where core_sub1 is not null order by
desig_value,emp_id ")
r=cursor.fetchall()
p=len(r)
c=0

morning=[301,301,301,301,301,301,301]
evening=[301,301,301,301,301,301,301]

while c<p:
    x=2
    f=4
    while (x!=0 and f<9):
```

```
        u=0
        a=[]
        g=0
        l=r[c][f]
        e="select * from course_details where course_code='%s' "%l
        cursor.execute(e)
        for i in cursor:
            a.append(i)
        q=len(a)
        s="select * from course_type where course_code='%s'"%l
        cursor.execute(s)
        for t in cursor:
            b=t
        s="select slot from allocation where emp_name='%s'"%r[c][0]
        cursor.execute(s)
        y=[]
        for o in cursor:
            y.append(o)
        l=0
        while (u!=q and g==0):
            if(len(y)>0):
                for i in range(0,len(y)):
                    if(a[u][2]==y[i][0]):
                        l=1
            if(a[u][3]>0 and l==0):
                g=1
                slot=str(a[u][2])
                s=slot
                if(slot=='A1' or slot=='B1' or slot=='C1' or slot=='D1' or slot=='E1' or slot=='F1' or
slot=='G1'):
```

```
if(s=='A1'):
    venue="SJT"+str(morning[0])
    morning[0]=morning[0]+1
    if(morning[0]%100 == 15):
        morning[0]=morning[0]+5
    if(morning[0]%100==25):
        morning[0]=(morning[0])+100-24
elif(s=='B1'):
    venue="SJT"+str(morning[1])
    morning[1]=morning[1]+1
    if(morning[1]%100 == 15):
        morning[1]=morning[1]+5
    if(morning[1]%100==25):
        morning[1]=(morning[1])+100-24
elif(s=='C1'):
    venue="SJT"+str(morning[2])
    morning[2]=morning[2]+1
    if(morning[2]%100 == 15):
        morning[2]=morning[2]+5
    if(morning[2]%100==25):
        morning[2]=(morning[2])+100-24
elif(s=='D1'):
    venue="SJT"+str(morning[3])
    morning[3]=morning[3]+1
    if(morning[3]%100 == 15):
        morning[3]=morning[0]+5
    if(morning[3]%100==25):
        morning[3]=(morning[3])+100-24
elif(s=='E1'):
```

```
            venue="SJT"+str(morning[4])
            morning[4]=morning[4]+1
            if(morning[4]%100 == 15):
                morning[4]=morning[4]+5
            if(morning[4]%100==25):
                morning[4]=(morning[4])+100-24
        elif(s=='F1'):
            venue="SJT"+str(morning[5])
            morning[5]=morning[5]+1
            if(morning[5]%100 == 15):
                morning[5]=morning[5]+5
            if(morning[5]%100==25):
                morning[5]=(morning[5])+100-24
        elif(s=='G1'):
            venue="SJT"+str(morning[6])
            morning[6]=morning[6]+1
            if(morning[6]%100 == 15):
                morning[6]=morning[6]+5
            if(morning[6]%100==25):
                morning[6]=(morning[6])+100-24
    else:
        if(s=='A2'):
            venue="SJT"+str(evening[0])
            evening[0]=evening[0]+1
            if(evening[0]%100 == 15):
                evening[0]=evening[0]+5
            if(evening[0]%100==25):
                evening[0]=(evening[0])+100-24
        elif(s=='B2'):
```

```
        venue="SJT"+str(evening[1])
        evening[1]=evening[1]+1
        if(evening[1]%100 == 15):
            evening[1]=evening[1]+5
        if(evening[1]%100==25):
            evening[1]=(evening[1])+100-24
    elif(s=='C2'):
        venue="SJT"+str(evening[2])
        evening[2]=evening[2]+1
        if(evening[2]%100 == 15):
            evening[2]=evening[2]+5
        if(evening[2]%100==25):
            evening[2]=(evening[2])+100-24
    elif(s=='D2'):
        venue="SJT"+str(evening[3])
        evening[3]=evening[3]+1
        if(evening[3]%100 == 15):
            evening[3]=evening[0]+5
        if(evening[3]%100==25):
            evening[3]=(evening[3])+100-24
    elif(s=='E2'):
        venue="SJT"+str(evening[4])
        evening[4]=evening[4]+1
        if(evening[4]%100 == 15):
            evening[4]=evening[4]+5
        if(evening[4]%100==25):
            evening[4]=(evening[4])+100-24
    elif(s=='F2'):
        venue="SJT"+str(evening[5])
```

```
                    evening[5]=evening[5]+1
                    if(evening[5]%100 == 15):
                        evening[5]=evening[5]+5
                    if(evening[5]%100==25):
                        evening[5]=(evening[5])+100-24
                elif(s=='G2'):
                    venue="SJT"+str(evening[6])
                    evening[6]=evening[6]+1
                    if(morning[6]%100 == 15):
                        evening[6]=evening[6]+5
                    if(evening[6]%100==25):
                        evening[6]=(evening[6])+100-24
            s="insert into
allocation(emp_name,emp_id,course_name,course_code,course_credits,slot,venue)
values('%s',%d,'%s','%s',%d,'%s','%s')"%(r[c][0],r[c][1],str(a[u][0]),str(a[u][1]),int(b[2]),str(a
[u][2]),venue)
            cursor.execute(s)
            s="update course_details set batches=batches-1 where course_code='%s' and
slot='%s'"%(l,r[c][2])
            cursor.execute(s)
            u=u+1
        else:
            u=u+1
        if(g==1):
            x=x-1
    f=f+1
    s="update teacher_wishlistc set ind=%d where emp_name = '%s'"%(f,r[c][0])
    cursor.execute(s)
  c=c+1
tt.commit()
#third subject allocation
```

```
print("15.Allocating max one more subject based of faculty priority....\n")
cursor.execute("select * from teacher_wishlistc where core_sub1 is not null order by
desig_value,emp_id ")
r=cursor.fetchall()
p=len(r)


c=0
while c<p:
    x=1
    cursor.execute("select ind from teacher_wishlistc where emp_name='%s'"%r[c][0])
    f=cursor.fetchone()[0]
    while (x!=0 and f<9):
        u=0
        a=[]
        g=0
        l=r[c][f]
        e="select * from course_details where course_code='%s' "%l
        cursor.execute(e)
        for i in cursor:
            a.append(i)
        q=len(a)
        s="select * from course_type where course_code='%s'"%l
        cursor.execute(s)
        for t in cursor:
            b=t
        s="select slot from allocation where emp_name='%s'"%r[c][0]
        cursor.execute(s)
        y=[]
        for o in cursor:
            y.append(o)
```

```
        l=0
        while (u!=q and g==0):
            if(len(y)>0):
                for i in range(0,len(y)):
                    if(a[u][2]==y[i][0]):
                        l=1
            if(a[u][3]>0 and l==0):
                g=1
                slot=str(a[u][2])
                s=slot
                if(slot=='A1' or slot=='B1' or slot=='C1' or slot=='D1' or slot=='E1' or slot=='F1' or
slot=='F1'):
                    if(s=='A1'):
                        venue="SJT"+str(morning[0])
                        morning[0]=morning[0]+1
                        if(morning[0]%100 == 15):
                            morning[0]=morning[0]+5
                        if(morning[0]%100==25):
                            morning[0]=(morning[0])+100-24
                    elif(s=='B1'):
                        venue="SJT"+str(morning[1])
                        morning[1]=morning[1]+1
                        if(morning[1]%100 == 15):
                            morning[1]=morning[1]+5
                        if(morning[1]%100==25):
                            morning[1]=(morning[1])+100-24
                    elif(s=='C1'):
                        venue="SJT"+str(morning[2])
                        morning[2]=morning[2]+1
                        if(morning[2]%100 == 15):
```

```
            morning[2]=morning[2]+5
        if(morning[2]%100==25):
            morning[2]=(morning[2])+100-24
    elif(s=='D1'):
        venue="SJT"+str(morning[3])
        morning[3]=morning[3]+1
        if(morning[3]%100 == 15):
            morning[3]=morning[0]+5
        if(morning[3]%100==25):
            morning[3]=(morning[3])+100-24
    elif(s=='E1'):
        venue="SJT"+str(morning[4])
        morning[4]=morning[4]+1
        if(morning[4]%100 == 15):
            morning[4]=morning[4]+5
        if(morning[4]%100==25):
            morning[4]=(morning[4])+100-24
    elif(s=='F1'):
        venue="SJT"+str(morning[5])
        morning[5]=morning[5]+1
        if(morning[5]%100 == 15):
            morning[5]=morning[5]+5
        if(morning[5]%100==25):
            morning[5]=(morning[5])+100-24
    elif(s=='G1'):
        venue="SJT"+str(morning[6])
        morning[6]=morning[6]+1
        if(morning[6]%100 == 15):
            morning[6]=morning[6]+5
```

```
            if(morning[6]%100==25):
                morning[6]=(morning[6])+100-24
        else:
            if(s=='A2'):
                venue="SJT"+str(evening[0])
                evening[0]=evening[0]+1
                if(evening[0]%100 == 15):
                    evening[0]=evening[0]+5
                if(evening[0]%100==25):
                    evening[0]=(evening[0])+100-24
            elif(s=='B2'):
                venue="SJT"+str(evening[1])
                evening[1]=evening[1]+1
                if(evening[1]%100 == 15):
                    evening[1]=evening[1]+5
                if(evening[1]%100==25):
                    evening[1]=(evening[1])+100-24
            elif(s=='C2'):
                venue="SJT"+str(evening[2])
                evening[2]=evening[2]+1
                if(evening[2]%100 == 15):
                    evening[2]=evening[2]+5
                if(evening[2]%100==25):
                    evening[2]=(evening[2])+100-24
            elif(s=='D2'):
                venue="SJT"+str(evening[3])
                evening[3]=evening[3]+1
                if(evening[3]%100 == 15):
                    evening[3]=evening[0]+5
```

```
                    if(evening[3]%100==25):
                        evening[3]=(evening[3])+100-24
                elif(s=='E2'):
                    venue="SJT"+str(evening[4])
                    evening[4]=evening[4]+1
                    if(evening[4]%100 == 15):
                        evening[4]=evening[4]+5
                    if(evening[4]%100==25):
                        evening[4]=(evening[4])+100-24
                elif(s=='F2'):
                    venue="SJT"+str(evening[5])
                    evening[5]=evening[5]+1
                    if(evening[5]%100 == 15):
                        evening[5]=evening[5]+5
                    if(evening[5]%100==25):
                        evening[5]=(evening[5])+100-24
                elif(s=='G2'):
                    venue="SJT"+str(evening[6])
                    evening[6]=evening[6]+1
                    if(morning[6]%100 == 15):
                        evening[6]=evening[6]+5
                    if(evening[6]%100==25):
                        evening[6]=(evening[6])+100-24
            s="insert into
allocation(emp_name,emp_id,course_name,course_code,course_credits,slot,venue)
values('%s',%d,'%s','%s',%d,'%s','%s')"%(r[c][0],r[c][1],str(a[u][0]),str(a[u][1]),int(b[2]),str(a
[u][2]),venue)
            cursor.execute(s)
            s="update course_details set batches=batches-1 where course_code='%s' and
slot='%s'"%(l,r[c][2])
            cursor.execute(s)
```

```
            u=u+1
        else:
            u=u+1
        if(g==1):
            x=x-1
      s="update teacher_wishlistc set ind=%d where emp_name = '%s'"%(f,r[c][0])
      cursor.execute(s)
      f=f+1
   c=c+1


tt.commit
#allocation for teachers having less than 3 subjects
print("16.Allocating remaining subjects to faculty with insufficient subjects...\n")
e="select * from course_details"
cursor.execute(e)
for i in cursor:
   a.append(i)
   q=len(a)
cursor.execute("select * from teacher_wishlistc where core_sub1 is not null order by
desig_value,emp_id ")
r=cursor.fetchall()
for c in range(0,len(r)):
   s="select * from allocation where emp_name='%s'"%r[c][0]
   cursor.execute(s)
   j=cursor.fetchall()
   x=3-len(j)
   k=0
   while (x!=0 and k!=q):
      #3 - course code
       f=0
```

```
vc=0
while(k<q and vc==0):
    s="select * from allocation where emp_name='%s'"%r[c][0]
    cursor.execute(s)
    j=cursor.fetchall()
    h=[]
    for w in range(0,len(j)):
        if(j[w][3]==a[k][1]):
            f==1
    if(f==0):
        s="select slot,course_code from allocation  where emp_name='%s'"%r[c][0]
        cursor.execute(s)
        y=[]
        for o in cursor:
            y.append(o)
        l=0
        if(len(y)>0):
            for i in range(0,len(y)):
                if(a[k][2]==y[i][0] or a[k][1]==y[i][1]):
                    l=1
        k=k+1
        if(l==0):
            h=a[k]
            vc=1
            s="select * from course_type where course_code='%s'"%h[1]
            cursor.execute(s)
            b=cursor.fetchone()
            if(h[3]>0 and l==0):
                slot=str(h[2])
```

```
                s=slot
                if(slot=='A1' or slot=='B1' or slot=='C1' or slot=='D1' or slot=='E1' or
slot=='F1' or slot=='F1'):
                    if(s=='A1'):
                        venue="SJT"+str(morning[0])
                        morning[0]=morning[0]+1
                        if(morning[0]%100 == 15):
                            morning[0]=morning[0]+5
                        if(morning[0]%100==25):
                            morning[0]=(morning[0])+100-24
                    elif(s=='B1'):
                        venue="SJT"+str(morning[1])
                        morning[1]=morning[1]+1
                        if(morning[1]%100 == 15):
                            morning[1]=morning[1]+5
                        if(morning[1]%100==25):
                            morning[1]=(morning[1])+100-24
                    elif(s=='C1'):
                        venue="SJT"+str(morning[2])
                        morning[2]=morning[2]+1
                        if(morning[2]%100 == 15):
                            morning[2]=morning[2]+5
                        if(morning[2]%100==25):
                            morning[2]=(morning[2])+100-24
                    elif(s=='D1'):
                        venue="SJT"+str(morning[3])
                        morning[3]=morning[3]+1
                        if(morning[3]%100 == 15):
                            morning[3]=morning[0]+5
                        if(morning[3]%100==25):
```

```
            morning[3]=(morning[3])+100-24
        elif(s=='E1'):
            venue="SJT"+str(morning[4])
            morning[4]=morning[4]+1
            if(morning[4]%100 == 15):
                morning[4]=morning[4]+5
            if(morning[4]%100==25):
                morning[4]=(morning[4])+100-24
        elif(s=='F1'):
            venue="SJT"+str(morning[5])
            morning[5]=morning[5]+1
            if(morning[5]%100 == 15):
                morning[5]=morning[5]+5
            if(morning[5]%100==25):
                morning[5]=(morning[5])+100-24
        elif(s=='G1'):
            venue="SJT"+str(morning[6])
            morning[6]=morning[6]+1
            if(morning[6]%100 == 15):
                morning[6]=morning[6]+5
            if(morning[6]%100==25):
                morning[6]=(morning[6])+100-24
    else:
        if(s=='A2'):
            venue="SJT"+str(evening[0])
            evening[0]=evening[0]+1
            if(evening[0]%100 == 15):
                evening[0]=evening[0]+5
            if(evening[0]%100==25):
```

```
            evening[0]=(evening[0])+100-24
    elif(s=='B2'):
        venue="SJT"+str(evening[1])
        evening[1]=evening[1]+1
        if(evening[1]%100 == 15):
            evening[1]=evening[1]+5
        if(evening[1]%100==25):
            evening[1]=(evening[1])+100-24
    elif(s=='C2'):
        venue="SJT"+str(evening[2])
        evening[2]=evening[2]+1
        if(evening[2]%100 == 15):
            evening[2]=evening[2]+5
        if(evening[2]%100==25):
            evening[2]=(evening[2])+100-24
    elif(s=='D2'):
        venue="SJT"+str(evening[3])
        evening[3]=evening[3]+1
        if(evening[3]%100 == 15):
            evening[3]=evening[0]+5
        if(evening[3]%100==25):
            evening[3]=(evening[3])+100-24
    elif(s=='E2'):
        venue="SJT"+str(evening[4])
        evening[4]=evening[4]+1
        if(evening[4]%100 == 15):
            evening[4]=evening[4]+5
        if(evening[4]%100==25):
            evening[4]=(evening[4])+100-24
```

```
                elif(s=='F2'):
                    venue="SJT"+str(evening[5])
                    evening[5]=evening[5]+1
                    if(evening[5]%100 == 15):
                        evening[5]=evening[5]+5
                    if(evening[5]%100==25):
                        evening[5]=(evening[5])+100-24
                elif(s=='G2'):
                    venue="SJT"+str(evening[6])
                    evening[6]=evening[6]+1
                    if(morning[6]%100 == 15):
                        evening[6]=evening[6]+5
                    if(evening[6]%100==25):
                        evening[6]=(evening[6])+100-24
            s="insert into
allocation(emp_name,emp_id,course_name,course_code,course_credits,slot,venue)
values('%s',%d,'%s','%s',%d,'%s','%s')"%(r[c][0],r[c][1],str(h[0]),str(h[1]),int(b[2]),str(h[2]),v
enue)
            cursor.execute(s)
            s="update course_details set batches=batches-1 where course_code='%s' and
slot='%s'"%(h[1],h[2])
            cursor.execute(s)
            x=x-1


print("17.Creating Final Allocation Excel sheet in DBMS PROJECT directory....\n")
allocation=Workbook()
sheet2=allocation.add_sheet('Allocation')
sheet2.col(0).width=6000
sheet2.col(2).width=12000
sheet2.col(3).width=5000
sheet2.col(1).width=4000
```

```
style=easyxf('align:horiz centre;pattern:pattern solid,fore_colour green;')
sheet2.write(0,0,'EMPLOYEE NAME',style)
sheet2.write(0,1,'EMPLOYEE ID',style)
sheet2.write(0,2,'COURSE NAME',style)
sheet2.write(0,3,'COURSE CODE',style)
sheet2.write(0,4,'CREDITS',style)
sheet2.write(0,5,'SLOT',style)
sheet2.write(0,6,'VENUE',style)

print("18.Exporting data from allocation table into allocation excel file....\n")
s="select * from allocation order by emp_name"
cursor.execute(s)
r=cursor.fetchall()
p=len(r)
x=0
c=1
style=easyxf('align:horiz centre;')

while(c<=p):
    x=0
    while(x<7):
        if(x==0 or x==2):
            sheet2.write(c,x,r[c-1][x])
        else:
            sheet2.write(c,x,r[c-1][x],style)
        x=x+1
    c=c+1
allocation.save('C:/Users/shreyansh/Desktop/DBMS PROJECT/allocation.xls')
```

```
print("19.Creating a 'Time Tables' directory inside DBMS PROJECT directory....\n")
try:
    os.mkdir("C:/Users/shreyansh/Desktop/DBMS PROJECT/Time Tables")
except OSError:
    pass
tt.commit()


print("20.Generating time table excel files for all the faculties based on allocation table and seniority of faculty....\n")
s="select emp_name from teacher_wishlist order by desig_value,emp_id"
cursor.execute(s)
r=cursor.fetchall()
for i in range(0,len(r)):
    time=Workbook()
    sheet=time.add_sheet('time table')
    style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom thin;pattern:pattern solid,fore_colour ice_blue;')
    sheet.write(0,0,'start',style)
    sheet.write(1,0,'end',style)
    style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom thin;pattern:pattern solid,fore_colour light_green;')
    sheet.write(2,0,'MON',style)
    sheet.write(3,0,'TUE',style)
    sheet.write(4,0,'WED',style)
    sheet.write(5,0,'THU',style)
    sheet.write(6,0,'FRI',style)
    for k in range(1,12):
        if(k!=6):
            sheet.col(k).width=5050
    style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom thin;pattern:pattern solid,fore_colour sky_blue;')
```

```
    sheet.write(0,1,'08:00',style)
    sheet.write(1,1,'08:50',style)
    sheet.write(0,2,'09:00',style)
    sheet.write(1,2,'09:50',style)
    sheet.write(0,3,'10:00',style)
    sheet.write(1,3,'10:50',style)
    sheet.write(0,4,'11:00',style)
    sheet.write(1,4,'11:50',style)
    sheet.write(0,5,'12:00',style)
    sheet.write(1,5,'12:50',style)
    sheet.write(0,7,'14:00',style)
    sheet.write(1,7,'14:50',style)
    sheet.write(0,8,'15:00',style)
    sheet.write(1,8,'15:50',style)
    sheet.write(0,9,'16:00',style)
    sheet.write(1,9,'16:50',style)
    sheet.write(0,10,'17:00',style)
    sheet.write(1,10,'17:50',style)
    sheet.write(0,11,'18:00',style)
    sheet.write(1,11,'18:50',style)
    sheet.write(0,12,'19:00',style)
    sheet.write(1,12,'19:50',style)
    style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom
thin;pattern:pattern solid,fore_colour light_yellow;')

    sheet.write(2,1,'A1',style)
    sheet.write(3,1,'B1',style)
    sheet.write(4,1,'C1',style)
    sheet.write(5,1,'D1',style)
```

```
    sheet.write(6,1,'E1',style)


    sheet.write(2,2,'F1',style)
    sheet.write(3,2,'G1',style)
    sheet.write(4,2,'A1',style)
    sheet.write(5,2,'B1',style)
    sheet.write(6,2,'C1',style)

    sheet.write(2,3,'D1',style)
    sheet.write(3,3,'E1',style)
    sheet.write(4,3,'F1',style)
    sheet.write(5,3,'G1',style)
    sheet.write(6,3,'TA1',style)

    sheet.write(2,4,'TB1',style)
    sheet.write(3,4,'TC1',style)
    sheet.write(4,4,'V1',style)
    sheet.write(5,4,'TE1',style)
    sheet.write(6,4,'TF1',style)

    sheet.write(2,5,'TG1',style)
    sheet.write(3,5,'TAA1',style)
    sheet.write(4,5,'V1',style)
    sheet.write(5,5,'TCC1',style)
    sheet.write(6,5,'TD1',style)

    style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom
thin;pattern:pattern solid,fore_colour gray_ega;')
    sheet.write(0,6,'Lunch',style)
    sheet.write(1,6,'Lunch',style)
```

```
    sheet.write(2,6,'L',style)
    sheet.write(3,6,'U',style)
    sheet.write(4,6,'N',style)
    sheet.write(5,6,'C',style)
    sheet.write(6,6,'H',style)

    style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom
thin;pattern:pattern solid,fore_colour light_yellow;')

    sheet.write(2,7,'A2',style)
    sheet.write(3,7,'B2',style)
    sheet.write(4,7,'C2',style)
    sheet.write(5,7,'D2',style)
    sheet.write(6,7,'E2',style)

    sheet.write(2,8,'F2',style)
    sheet.write(3,8,'G2',style)
    sheet.write(4,8,'A2',style)
    sheet.write(5,8,'B2',style)
    sheet.write(6,8,'C2',style)

    sheet.write(2,9,'D2',style)
    sheet.write(3,9,'E2',style)
    sheet.write(4,9,'F2',style)
    sheet.write(5,9,'G2',style)
    sheet.write(6,9,'TA2',style)

    sheet.write(2,10,'TB2',style)
    sheet.write(3,10,'TC2',style)
    sheet.write(4,10,'V2',style)
```

```
sheet.write(5,10,'TE2',style)
sheet.write(6,10,'TF2',style)

sheet.write(2,11,'TG2',style)
sheet.write(3,11,'TAA2',style)
sheet.write(4,11,'V2',style)
sheet.write(5,11,'TCC2',style)
sheet.write(6,11,'TD2',style)

sheet.write(2,12,'V3',style)
sheet.write(3,12,'V4',style)
sheet.write(4,12,'V5',style)
sheet.write(5,12,'V6',style)
sheet.write(6,12,'V7',style)
s="select slot,course_code,venue from allocation where emp_name='%s'"%r[i][0]
cursor.execute(s)
f=cursor.fetchall()
p=len(f)
dot="."
v=str(str(i+1)+dot+r[i][0])
t=("C:/Users/shreyansh/Desktop/DBMS PROJECT/time tables/%s.xls")%v
time.save(t)
t=("C:/Users/shreyansh/Desktop/DBMS PROJECT/time tables/%s.xls")%v
rb = xlrd.open_workbook(t,formatting_info=True)
r_sheet = rb.sheet_by_index(0)
wb =copy(rb)
sheet = wb.get_sheet(0)
for j in range(0,p):
    #print(f[j][0],f[j][1],f[j][2])
```

```
        style=easyxf('align:horiz centre;borders: left thin, right thin, top thin, bottom thin;pattern:pattern solid,fore_colour bright_green;')

        if(f[j][0]=='A1' or f[j][0]=='B1' or f[j][0]=='C1' or f[j][0]=='D1' or f[j][0]=='E1' or f[j][0]=='F1' or f[j][0]=='G1' ):

            if(f[j][0]=='A1'):

                s=("A1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(2,1,s,style)

                s=("A1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(4,2,s,style)

                s=("TA1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(6,3,s,style)

                s=("TAA1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(3,5,s,style)

            if(f[j][0]=='B1'):

                s=("B1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(3,1,s,style)

                s=("B1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(5,2,s,style)

                s=("TB1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(2,4,s,style)

            if(f[j][0]=='C1'):

                s=("C1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(4,1,s,style)

                s=("C1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(6,2,s,style)

                s=("TC1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(3,4,s,style)

                s=("TCC1 \n%s %s")%(f[j][1],f[j][2])

                sheet.write(5,5,s,style)

            if(f[j][0]=='D1'):
```

```
            s=("D1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(5,1,s,style)
            s=("D1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(2,3,s,style)
            s=("TD1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(6,5,s,style)
        if(f[j][0]=='E1'):
            s=("E1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(6,1,s,style)
            s=("E1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(3,3,s,style)
            s=("TE1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(5,4,s,style)
        if(f[j][0]=='F1'):
            s=("F1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(2,2,s,style)
            s=("F1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(4,3,s,style)
            s=("TF1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(6,4,s,style)
        if(f[j][0]=='G1'):
            s=("G1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(3,2,s,style)
            s=("G1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(5,3,s,style)
            s=("TG1 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(2,5,s,style)
    else:
        if(f[j][0]=='A2'):
```

```
        s=("A2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(2,7,s,style)
        s=("A2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(4,8,s,style)
        s=("TA2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(6,9,s,style)
        s=("TAA2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(3,11,s,style)
    if(f[j][0]=='B2'):
        s=("B2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(3,7,s,style)
        s=("B2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(5,8,s,style)
        s=("TB2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(2,10,s,style)
    if(f[j][0]=='C2'):
        s=("C2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(4,7,s,style)
        s=("C2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(6,8,s,style)
        s=("TC2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(3,10,s,style)
        s=("TCC2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(5,11,s,style)
    if(f[j][0]=='D2'):
        s=("D2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(5,7,s,style)
        s=("D2 \n%s %s")%(f[j][1],f[j][2])
        sheet.write(2,9,s,style)
```

```
            s=("TD2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(6,11,s,style)
        if(f[j][0]=='E2'):
            s=("E2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(6,7,s,style)
            s=("E2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(3,9,s,style)
            s=("TE2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(5,10,s,style)
        if(f[j][0]=='F1'):
            s=("F2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(2,8,s,style)
            s=("F2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(4,9,s,style)
            s=("TF2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(6,10,s,style)
        if(f[j][0]=='G2'):
            s=("G2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(3,8,s,style)
            s=("G2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(5,9,s,style)
            s=("TG2 \n%s %s")%(f[j][1],f[j][2])
            sheet.write(2,11,s,style)
    dot="."
    v=str(str(i+1)+dot+r[i][0])
    t=("C:/Users/shreyansh/Desktop/DBMS PROJECT/time tables/%s.xls")%v
    wb.save(t)
print("21.All time tables genereted....\n")
print("Dynamic Time Table Generation Complete\n\n")
```

```
print("\t\t\tTHANK YOU FOR USING OUR SERVIES")
print("Press anything to exit")
x=input()
```

6. CONCLUSION

A folder is created on the desktop which contains two excel sheets, one for the allocation list and the other for teacher wishlist. The folder also contains a directory that has separate timetable for the individual faculty are generated automatically by this system. Various slot combinations can be acquired so that another timetable is generated as of need. The project reduces time consumption. The project is developed in such a way that, no slot clashes occur providing features to tailor the timetable as of wish. Additional features that is included in the project is that core as well as elective subjects are allocated to the faculty. Also, the priority of the Wishlist generated is given by the designation of the faculty that is senior faculty is given higher priority over lower priority teacher also the GUI systems enables a faculty to sign up and login and then select at max 5 subjects where there are 2 core and 3 electives.

7. REFERENCES

https://www.youtube.com/watch?v=PSm-tq5M-Dc&index=9&list=PL6gx4Cwl9DGBwibXFtPtflztSNPGuIB_d

https://www.youtube.com/watch?v=PSm-tq5M-Dc&index=9&list=PL6gx4Cwl9DGBwibXFtPtflztSNPGuIB_d

https://www.geeksforgeeks.org/global-local-variables-python/

https://www.w3schools.com/python/python_mysql_getstarted.asp